IN THE CLAIMS

Please amend the claims as follows.

1. (currently amended) A method for ~~detecting cross-iteration dependencies between variables in a loop of a computer program, the method comprising the steps of:~~ executing, by a processor of a computer system, a set of program instructions for a loop, wherein the executing comprises performing the steps of:

storing in a tangible, computer-readable storage media a set of unique proxy values, the ~~associating~~ unique proxy values being substituted for ~~with each of the~~ respective values of indirect loop index variables of the loop;

calculating by the processor, ~~for each iteration of the loop an~~ wherein a result of the calculating is indirectly indexed access patterns for respective iterations of the loop based upon the ~~associated~~ unique proxy values for the indirect loop index variables, wherein each proxy value is a unique prime number and the resulting indirectly indexed access patterns have respective numbers of pattern values, and wherein none of the respective numbers of pattern values exceeds three regardless of how many statements are in the loop; and

executing the set of program instructions for the loop by the processor, wherein ones of the iterations are executed concurrently responsive to determining that no ~~whether~~ cross-iteration dependencies exist between the ones of the ~~any two~~ iterations of the loop based upon the indirectly indexed access patterns of the ones of the ~~two~~ iterations.

2-5. (canceled)

6. (currently amended) The method as claimed in claim 1, ~~-5,~~ wherein the ~~indirectly indexed access~~ pattern values for respective ~~an~~ iterations are ~~is~~ calculated by forming ~~the~~ products of ~~the unique prime numbers associated with each of~~ the proxy values of the indirect loop index variables of the loop for the respective ~~that~~ iterations.

7. (currently amended) The method as claimed in claim 6, wherein the determining indicates existence of a cross-iteration dependency~~ies~~ ~~if is determined by determining whether~~ a

greatest common divisor between ~~the~~ two indirectly indexed access patterns for ~~the~~ two respective ~~corresponding~~ iterations is greater than one.

8-17. (canceled)

17. A computer program product for ~~detecting cross-iteration dependencies between variables in a loop of a computer program,~~ executing, by a processor of a computer system, a set of program instructions for a loop, the computer program product comprising computer software stored on a computer-readable medium for performing the steps of:

storing in a tangible, computer-readable storage media a set of unique proxy values, the ~~associating~~ unique proxy values being substituted for ~~with each of the~~ respective values of indirect loop index variables of the loop;

calculating by the processor, ~~for each iteration of the loop an~~ wherein a result of the calculating is indirectly indexed access patterns for respective iterations of the loop based upon the ~~associated~~ unique proxy values for the indirect loop index variables, wherein each proxy value is a unique prime number and the resulting indirectly indexed access patterns have respective numbers of pattern values, and wherein none of the respective numbers of pattern values exceeds three regardless of how many statements are in the loop; and

executing the set of program instructions for the loop by the processor, wherein ones of the iterations are executed concurrently responsive to determining that no ~~whether~~ cross-iteration dependencies exist between the ones of the ~~any two~~ iterations of the loop based upon the indirectly indexed access patterns of the ones of the ~~two~~ iterations.

18. A computer system ~~for detecting cross-iteration dependencies between variables in a loop of a computer program, the computer system~~ executing program instructions ~~computer software~~ stored on a computer-readable medium for ~~performing the steps of:~~ executing, by a processor of the computer system, a set of the program instructions for a loop, wherein the executing comprises performing the steps of:

storing in a tangible, computer-readable storage media a set of unique proxy values, the ~~associating~~ unique proxy values being substituted for ~~with each of the~~ respective values of indirect loop index variables of the loop;

calculating by the processor, ~~for each iteration of the loop an~~ wherein a result of the calculating is indirectly indexed access patterns based upon the ~~associated~~ unique proxy values for the indirect loop index variables, wherein each proxy value is a unique prime number and the resulting indirectly indexed access patterns have respective numbers of pattern values, and wherein none of the respective numbers of pattern values exceeds three regardless of how many statements are in the loop; and

executing the set of program instructions for the loop by the processor, wherein ones of the iterations are executed concurrently responsive to determining that no ~~whether~~ cross-iteration dependencies exist between the ones of the ~~any two~~ iterations of the loop based upon the indirectly indexed access patterns of the ones of the ~~two~~ iterations.

19. (new) The method of claim 1, wherein if the loop includes a boolean condition for selecting active and inactive statements defined in assignment statements of the loop body, such an indirectly indexed access pattern for a respective one of the iterations consists of i) a first pattern value generated responsive to indirect loop indices of array variables for such statements active if the boolean condition evaluates to be true and responsive to indirect loop indices of array variables for such statements active if the boolean condition evaluates to be false, ii) a second pattern value generated responsive to indirect loop indices of array variables for such statements active if the boolean condition evaluates to be true, and iii) a third pattern value generated responsive to indirect loop indices of array variables for such statements active if the boolean condition evaluates to be false, and wherein if the loop does not include a boolean condition for selecting active and inactive statements defined in assignment statements of the loop body, the indirectly indexed access pattern consists of a single pattern value generated responsive to indirect loop indices of all array variables defined in assignment statements of the loop body.

20. (new) The computer program product of claim 17, wherein if the loop includes a boolean condition for selecting active and inactive statements defined in assignment statements of the loop body, such an indirectly indexed access pattern for a respective one of the iterations consists of i) a first pattern value generated responsive to indirect loop indices of array variables for such statements active if the boolean condition evaluates to be true and responsive to indirect loop indices of array variables for such statements active if the boolean condition evaluates to be false, ii) a second pattern value generated responsive to indirect loop indices of array variables for such statements active if the boolean condition evaluates to be true, and iii) a third pattern value generated responsive to indirect loop indices of array variables for such statements active if the boolean condition evaluates to be false, and wherein if the loop does not include a boolean condition for selecting active and inactive statements defined in assignment statements of the loop body, the indirectly indexed access pattern consists of a single pattern value generated responsive to indirect loop indices of all array variables defined in assignment statements of the loop body.

21. (new) The system of claim 17, wherein if the loop includes a boolean condition for selecting active and inactive statements defined in assignment statements of the loop body, such an indirectly indexed access pattern for a respective one of the iterations consists of i) a first pattern value generated responsive to indirect loop indices of array variables for such statements active if the boolean condition evaluates to be true and responsive to indirect loop indices of array variables for such statements active if the boolean condition evaluates to be false, ii) a second pattern value generated responsive to indirect loop indices of array variables for such statements active if the boolean condition evaluates to be true, and iii) a third pattern value generated responsive to indirect loop indices of array variables for such statements active if the boolean condition evaluates to be false, and wherein if the loop does not include a boolean condition for selecting active and inactive statements defined in assignment statements of the loop body, the indirectly indexed access pattern consists of a single pattern value generated responsive to indirect loop indices of all array variables defined in assignment statements of the loop body.

22. (new) The computer program product as claimed in claim 17, wherein the pattern values for respective iterations are calculated by forming products of the proxy values of the indirect loop index variables of the loop for the respective iterations.

23. (new) The computer program product as claimed in claim 22, wherein the determining indicates existence of a cross-iteration dependency if a greatest common divisor between two indirectly indexed access patterns for two respective iterations is greater than one.

24. (new) The system as claimed in claim 18, wherein the pattern values for respective iterations are calculated by forming products of the proxy values of the indirect loop index variables of the loop for the respective iterations.

25. (new) The system as claimed in claim 24, wherein the determining indicates existence of a cross-iteration dependency if a greatest common divisor between two indirectly indexed access patterns for two respective iterations is greater than one.